LOGIN FORM TESTING

Automated Regression Testing and Report Generation for Login Functionality using Selenium, Cucumber & Allure

Under the guidance of :
Dr. SURYAKIRAN CHEBROLU

Submitted by:

Reg No	Name
2300030176	D.TRIVENI
2300033251	K.HARSHITHA
2300031595	T.BHUVANESHWARI
2300033498	RAVI SRAVANTHI

TABLE OF CONTENTS

ABSTRACT

Login forms are among the most common and critical components of modern software applications, serving as the primary entry point for users to access secure systems such as online banking, e-commerce platforms, social networking sites, and enterprise solutions. Inadequate testing of the login module can lead to unauthorized access, security vulnerabilities, and poor user experience, emphasizing the need for systematic verification and validation.

This project focuses on designing a comprehensive automated testing framework for a sample login page, covering functional validation, boundary value analysis, and basic security checks. The framework evaluates scenarios including valid and invalid credentials, empty input fields, excessive input lengths, special character handling, and verification of meaningful error messages. Black-box testing techniques are employed to simulate real user interactions without relying on internal system knowledge.

Security-oriented validations such as SQL injection attempts and password strength checks are also incorporated. The framework is implemented using Selenium WebDriver for browser automation and JUnit for structured test execution. Parameterized test data improves coverage across multiple scenarios, while reporting tools such as Allure Reports and HTML summaries provide clear visualization of execution results and defect analysis.

The outcome of this project includes a reusable set of automated login test cases, detailed execution reports, and enhanced assurance of login reliability and robustness. Given the universal presence of login systems, this framework can be applied across various domains including finance, healthcare, e-learning, and enterprise applications.

PROBLEM STATEMENT

Modern web applications rely heavily on login modules to control user access and protect sensitive information. Any defect in the login functionality may lead to serious consequences such as unauthorized access, compromised security, system misuse, or poor user experience. Manual testing of login forms is repetitive, time-consuming, and susceptible to human errors, especially when validating multiple input combinations and boundary conditions.

Additionally, frequent updates or enhancements to software systems may unintentionally introduce defects into previously working features. Therefore, a systematic regression testing approach is required to ensure that the login mechanism continues to function correctly after modifications.

To address these challenges, an automated testing framework is needed to validate login behavior under various scenarios, including valid and invalid credentials, empty fields, boundary value inputs, and security-related cases. Automation helps improve test coverage, accuracy, efficiency, and repeatability while enabling the generation of professional execution reports for analysis.

OBJECTIVES

The primary objectives of this project are to design and implement an automated testing framework for validating the login functionality of a web application. The project aims to ensure that the authentication mechanism correctly accepts valid credentials and rejects invalid inputs under various testing conditions.

Another objective is to perform comprehensive functional testing by verifying different login scenarios, including successful login, invalid username, invalid password, and combinations of incorrect credentials. The framework also focuses on validating mandatory field behavior by testing empty username and password inputs.

The project further aims to apply boundary value analysis to evaluate system behavior when input fields receive minimum, maximum, and excessive character lengths. Additional objectives include validating input variations such as case sensitivity, special characters, and numeric values.

A key objective of this work is to improve testing efficiency, accuracy, and repeatability by replacing manual testing with Selenium-based automation.

The framework is designed following the Page Object Model (POM) to enhance maintainability and scalability.

Finally, the project seeks to generate detailed execution reports using Allure Reporting, providing clear insights into pass/fail status, step-wise execution details, and defect identification.

METHODOLOGY

Testing Approach

The testing strategy adopted for this project is based on the principles of Black-Box Testing. In this approach, the login functionality of the application is evaluated by analyzing system behavior through inputs and outputs without considering the internal implementation details. The system is tested from an end-user perspective to ensure correctness and reliability.

Testing Techniques Applied

Functional Testing:
Functional testing was conducted to verify whether the login module performs according to the specified requirements. Both valid and invalid credential combinations were tested to validate expected outcomes.

Regression Testing:
Regression testing was performed to confirm that existing login features continue to function correctly after executing multiple test scenarios. This ensures that no unintended defects are introduced.

Boundary Value Analysis (BVA):
Boundary value testing was applied to assess the system’s behavior at the limits of input constraints. Test cases were designed to evaluate minimum, maximum, and excessive input lengths.

Negative Testing:
Negative testing focused on validating system responses to incorrect, incomplete, or unexpected inputs, such as invalid credentials and empty fields.

Basic Security Validation:
Security-oriented checks, including SQL injection attempts and improper input handling, were considered to evaluate the robustness of the login mechanism.

TOOLS AND TECHNOLOGIES

The automation testing framework developed in this project utilizes a combination of industry-standard tools and technologies to ensure efficient test execution, maintainability, and accurate reporting. Each component plays a specific role within the automation ecosystem.

Selenium WebDriver:
Selenium WebDriver is used as the primary browser automation tool. It enables interaction with web elements, execution of user actions such as entering credentials and clicking buttons, and validation of application behavior across real browsers. Selenium provides flexibility, cross-browser compatibility, and robust support for UI-based functional testing.

Java:
Java serves as the core programming language for implementing the automation logic. It offers platform independence, strong object-oriented features, and extensive library support. Java enables the creation of reusable methods, structured test steps, and integration with testing frameworks and build tools.

Cucumber (Behavior-Driven Development):
Cucumber is employed to implement Behavior-Driven Development (BDD), allowing test scenarios to be written in a human-readable Gherkin format. This improves collaboration between technical and non-technical stakeholders by expressing test cases as business-level requirements. Cucumber also facilitates data-driven testing through Scenario Outlines.

JUnit:
JUnit is used for managing test execution and performing validations. It provides assertion mechanisms to compare expected and actual outcomes, ensuring correctness of login behavior. JUnit supports structured test lifecycle management and seamless integration with Maven.

Maven:
Maven is utilized as the project build and dependency management tool. It simplifies library integration, version control, and execution of the test suite using standardized

commands. Maven ensures consistency, reproducibility, and automation of the build process.

WebDriverManager:
WebDriverManager automates the management of browser driver binaries. It eliminates the need for manual driver downloads and configuration by dynamically resolving compatible driver versions. This enhances portability and reduces environment setup issues.

Allure Reports:
Allure Reports is integrated to generate advanced, interactive execution reports. It provides detailed visualization of test outcomes, pass/fail statistics, execution timelines, and failure diagnostics. Allure enhances defect analysis and improves presentation quality.

TEST EXECUTION

Test execution is a critical phase in the software testing lifecycle where the designed test cases are run to validate the behavior of the application. In this project, automated test scripts were executed to verify the functionality, validation rules, and robustness of the login module.

The execution process began with launching the web browser using Selenium WebDriver. The automation framework navigated to the login page of the selected application and performed user-like interactions such as entering usernames, passwords, and submitting the login form. Each test scenario was executed based on predefined input combinations specified within the Cucumber feature file.

The implemented test cases covered multiple validation dimensions, including:

- Valid Login Scenarios: Verification of successful authentication using correct credentials.
- Invalid Login Scenarios: Handling of incorrect username and password combinations.
- Boundary Value Testing: Testing input limits such as minimum and maximum password lengths.
- Empty Field Validation: Ensuring proper error messages when mandatory fields are left blank.
- Special Character Handling: Validation of inputs containing symbols and non-alphanumeric characters.
- Security-Oriented Scenarios: Testing resilience against invalid input patterns.

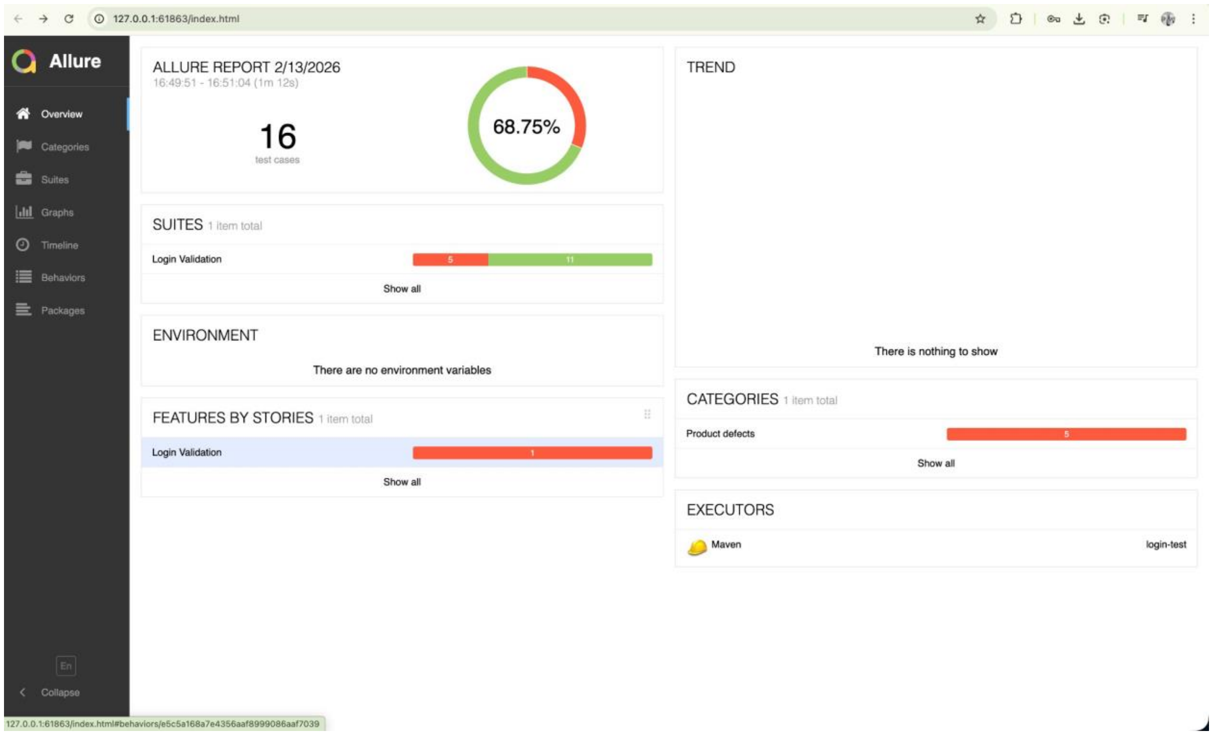
During execution, the system behavior was continuously monitored. Assertions implemented using JUnit validated whether the actual outcome matched the expected result. Test outcomes were recorded automatically, including pass/fail status and error details in case of failures.

The automation framework also captured execution artifacts such as logs and reports. HTML-based Cucumber reports and Allure Reports were generated to provide structured visualization of results. These reports helped in identifying failed scenarios, understanding defect patterns, and evaluating system stability.

RESULTS

The following screenshots illustrate the automated test execution results generated during the testing process.

Allure Report Overview:


127.0.0.1:61863/index.html
Allure
Overview
Categories
Suites
Graphs
Timeline
Behaviors
Packages
En
Collapse
127.0.0.1:61863/index.html#behaviors/e5c5a168a7e4356aaf8999086aaf7039

ALLURE REPORT 2/13/2026
16:49:51 - 16:51:04 (1m 12s)
16
test cases
68.75%

SUITES 1 item total
Login Validation 5 11
Show all

ENVIRONMENT
There are no environment variables

FEATURES BY STORIES 1 item total
Login Validation 1
Show all

TREND
There is nothing to show

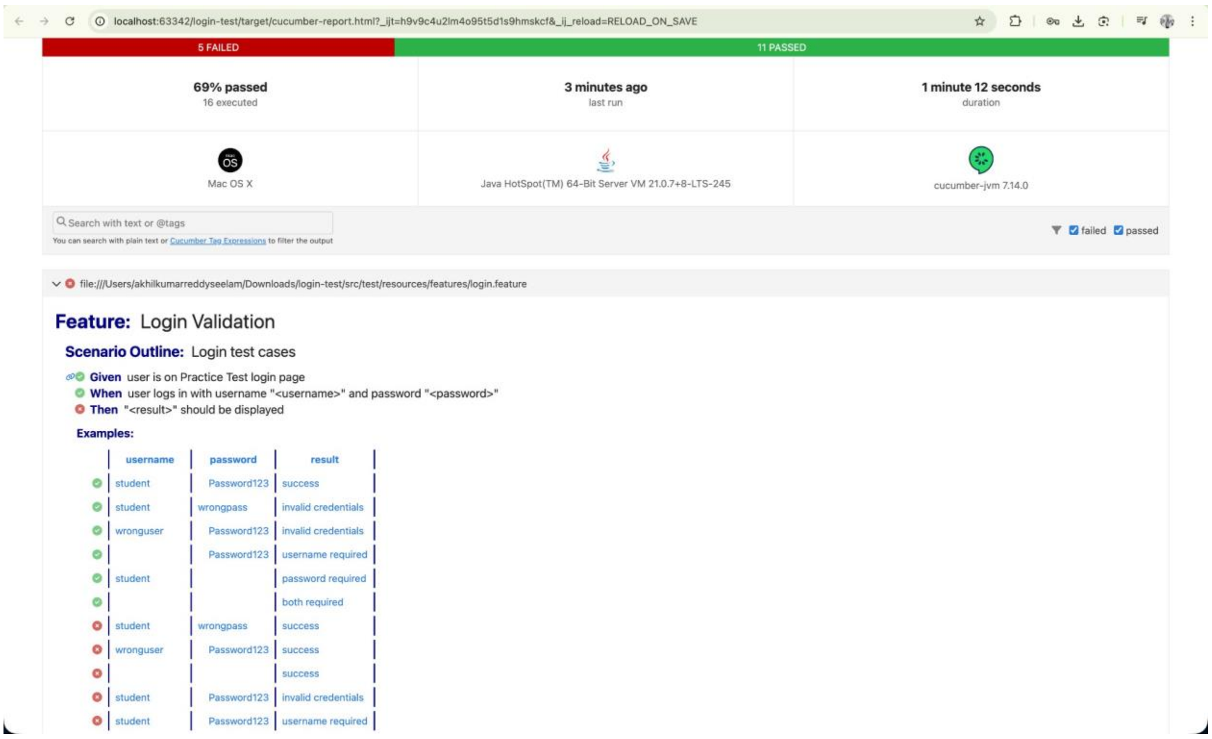
CATEGORIES 1 item total
Product defects 5
Show all

EXECUTORS
Maven login-test

Figure:Allure Report – Execution Summary

The Allure Report provides a graphical representation of test execution status, including passed and failed scenarios, execution duration, and detailed step logs.

Cucumber HTML Report



localhost:63342/login-test/target/cucumber-report.html?_ijt=h9v9c4u2lm4o95t5d1s9hmskcf&_ij_reload=RELOAD_ON_SAVE

5 FAILED | 11 PASSED

69% passed 16 executed	**3 minutes ago** last run	**1 minute 12 seconds** duration
Mac OS X	Java HotSpot(TM) 64-Bit Server VM 21.0.7+8-LTS-245	cucumber-jvm 7.14.0

Search with text or @tags

You can search with plain text or Cucumber Tag Expressions to filter the output

✓ failed ✓ passed

file:///Users/akhilkumarreddyseelam/Downloads/login-test/src/test/resources/features/login.feature

Feature: Login Validation

Scenario Outline: Login test cases

✓ **Given** user is on Practice Test login page

✓ **When** user logs in with username "<username>" and password "<password>"

✗ **Then** "<result>" should be displayed

Examples:

	username	password	result
✓	student	Password123	success
✓	student	wrongpass	invalid credentials
✓	wronguser	Password123	invalid credentials
✓		Password123	username required
✓	student		password required
✓			both required
✗	student	wrongpass	success
✗	wronguser	Password123	success
✗			success
✗	student	Password123	invalid credentials
✗	student	Password123	username required

Cucumber HTML Report – Scenario Results

The Cucumber HTML report displays scenario-wise execution results, highlighting step outcomes and validation status.

CONCLUSION

This project successfully demonstrated the design and implementation of an automated testing framework for validating the functionality and reliability of a web-based login form. The framework was developed using Selenium WebDriver, Cucumber, JUnit, and Maven, enabling structured test execution and efficient result reporting.

A comprehensive suite of automated test cases was created to evaluate multiple login scenarios, including valid authentication, invalid credentials, empty field validation, boundary value conditions, and negative test situations. The execution results confirmed that the login module behaved as expected under diverse input combinations while enforcing validation rules and displaying appropriate system responses.

The integration of reporting tools such as Cucumber HTML Reports and Allure Reports provided clear visualization of pass/fail outcomes, execution steps, and defect indications. These reports enhanced the transparency of the testing process and facilitated easier analysis of test results.

The adoption of automation significantly reduced manual effort, improved repeatability, and ensured consistency in test execution. The modular framework design, based on the Page Object Model, supports maintainability and scalability for future enhancements.

In conclusion, the project validates that automated testing is an effective approach for verifying login functionalities, improving application quality, and ensuring reliable user authentication mechanisms. The developed framework can be extended to other modules and applications requiring systematic validation.